SERVER ARCHITECTURES

1. Using Threads – threading library (Python) and pthreads C langauge:

- Define a main event loop thread which accepts connections
- After connection spawn a new thread to handle the connection.

2. Uisng Processes – multiprocessing (Python) and fork C langauge:

- Define a main event loop process which accepts connections
- After connection for a new process to handle the connection.

3. Pre-forking and Pre-Process:

- Create a pool of threads or processes and selects from this pool when a connection arrives.
- Libraries – concurrent.futures in Python is an abstraction designed to handle this situation
- I'm not sure if C has the abstraction or the pool must be created manually and managed as such.

4. I/O multiplexing – Python selectors (Library) and C (select and poll system calls or their corresponding C library wrappers):

- Create an event loop that polls or selects the available sockets and then invokes the corresponding callback function.
- It's a pure event driven architecture where events are received and passed through to the corresponding callback without blocking the main loop which listens for events.

5. Asyncronous programming – Python library (asyncio), C not sure yet:

- Uses async and await keywords
- Doesn't block the main function composed of various tasks. When a task runs and encouters await, control is passed to other tasks. Avoids blocking.